Topics in Reinforcement Learning: Lessons from AlphaZero for (Sub)Optimal Control and Discrete Optimization

Arizona State University
Course CSE 691, Spring 2023

Links to Class Notes, Videolectures, and Slides at
http://web.mit.edu/dimitrib/www/RLbook.html

Dimitri P. Bertsekas
dbertsek@asu.edu

Lecture 3
Linear Quadratic Problems, Approximation in Value Space, and Newton's Method
Problem Formulations, Reformulations, and Examples

Outline

Solution of the Deterministic (One-D) Linear Quadratic Problem

- System: $x_{k+1} = ax_k + bu_k$. Cost function: $\lim_{N\to\infty} \sum_{k=0}^{N-1} (qx_k^2 + ru_k^2)$
- The min-Bellman eq. is $J(x) = \min_u \left[qx^2 + ru^2 + J(ax + bu)\right]$
- For linear $\mu(x) = Lx$, the L-Bellman eq. is $J(x) = (q + rL^2)x^2 + J\big((a + bL)x\big)$
- We try quadratic solutions, $J(x) = Kx^2$, to Bellman eqs. and obtain Riccati eqs. (after we cancel x^2)

$$K = F(K) = \frac{a^2 rK}{r + b^2 K} + q, \qquad K = F_L(K) = (a + bL)^2 K + q + rL^2$$

- If $K^* > 0$ solves min-Riccati eq., then $J^*(x) = K^* x^2$, and we obtain the optimal policy from

$$\mu^*(x) = \arg\min_u \left[qx^2 + ru^2 + K^*(ax + bu)^2\right].$$

It is the linear function of x, $\mu^*(x) = L^* x$, with

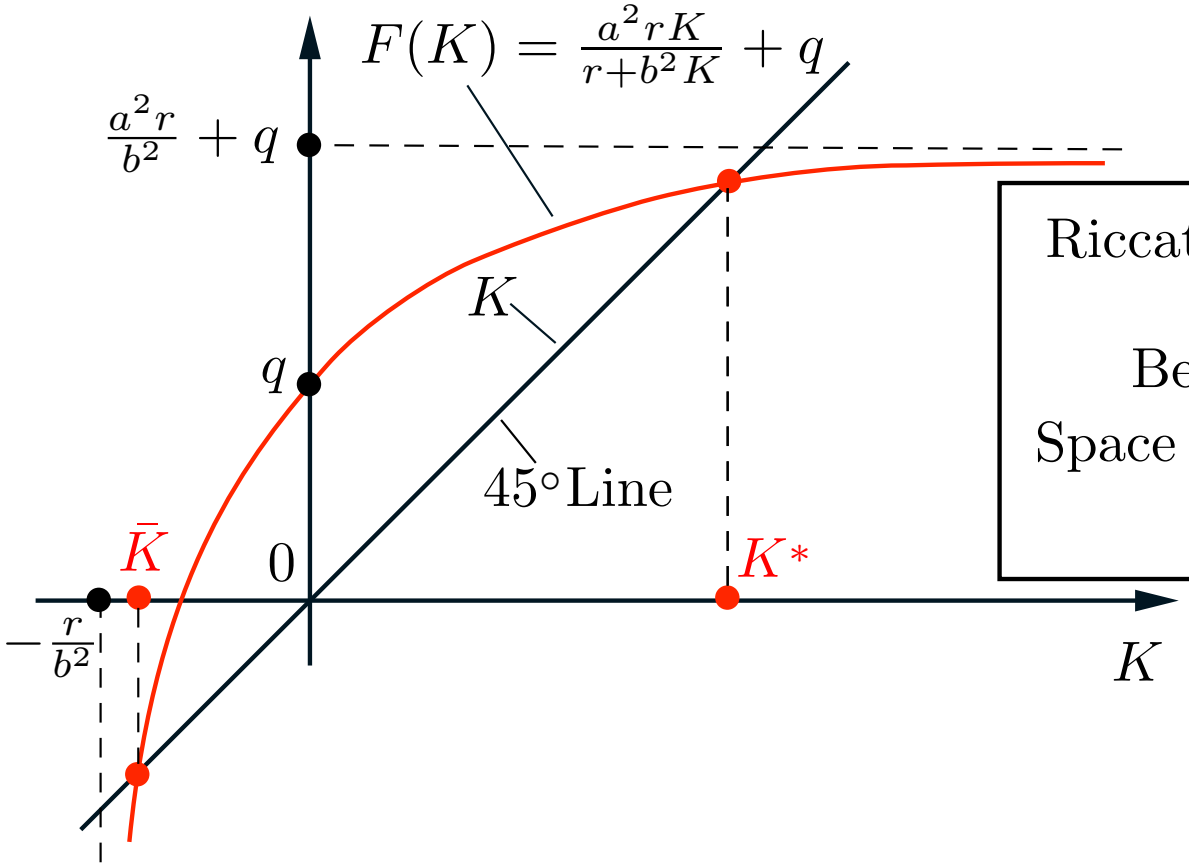
$$L^* = -\frac{abK^*}{r + b^2 K^*}$$

- Starting with quadratic $J_0(x) = K_0 x^2$, the VI iterates are quadratic: $J_k(x) = K_k x^2$, where $\{K_k\}$ is generated by

$$K_{k+1} = F(K_k) = \frac{a^2 rK_k}{r + b^2 K_k} + q$$

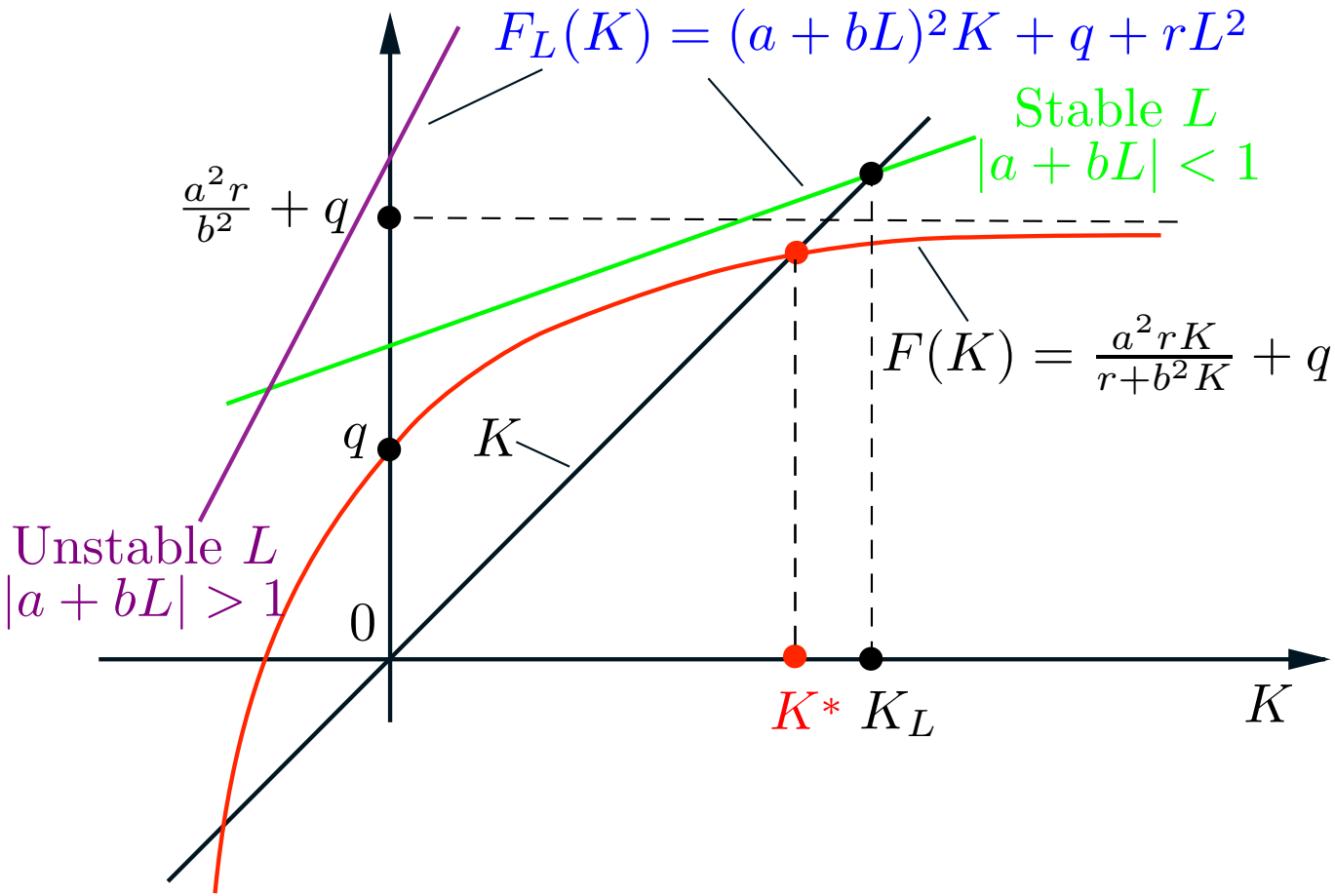
- Starting with a linear policy $\mu^0 = L_0 x$, the PI iterates are linear: $\mu^k = L_k x$

Graphical Solution of the Min-Riccati Equation



Riccati Equation: $K = F(K)$
from
Bellman Equation on
Space of Quadratic Functions
$J(x) = Kx^2$

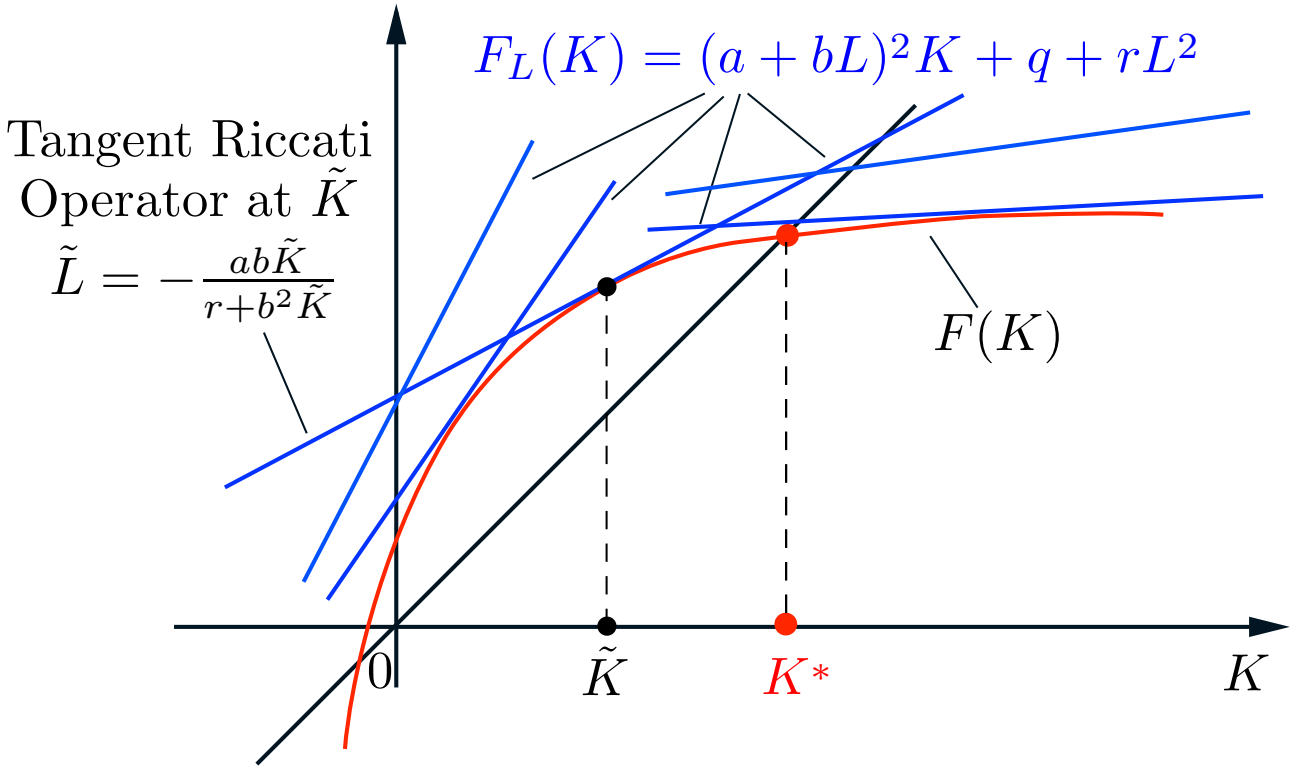
Relation of Min-Riccati Equation and L-Riccati Equations



RICCATI EQUATIONS $K^* = F(K^*)$, $K_L = F_L(K_L)$

- For $\mu(x) = Lx$ with $|a+bL| < 1$, the closed-loop linear system $x_{k+1} = (a+bL)x_k$ is stable, and we have $J_\mu(x) = K_L x^2$, where K_L solves the L-Riccati equation
- For $\mu(x) = Lx$ with $|a+bL| > 1$, the closed-loop linear system is unstable, and we have $J_\mu(x) = \infty$ for all $x \neq 0$

Min-Riccati Operator as Lower Envelope of L-Riccati Operators



$$F(K)x^2 = \min_{u\in\Re}\left\{qx^2 + ru^2 + K(ax+bu)^2\right\}$$

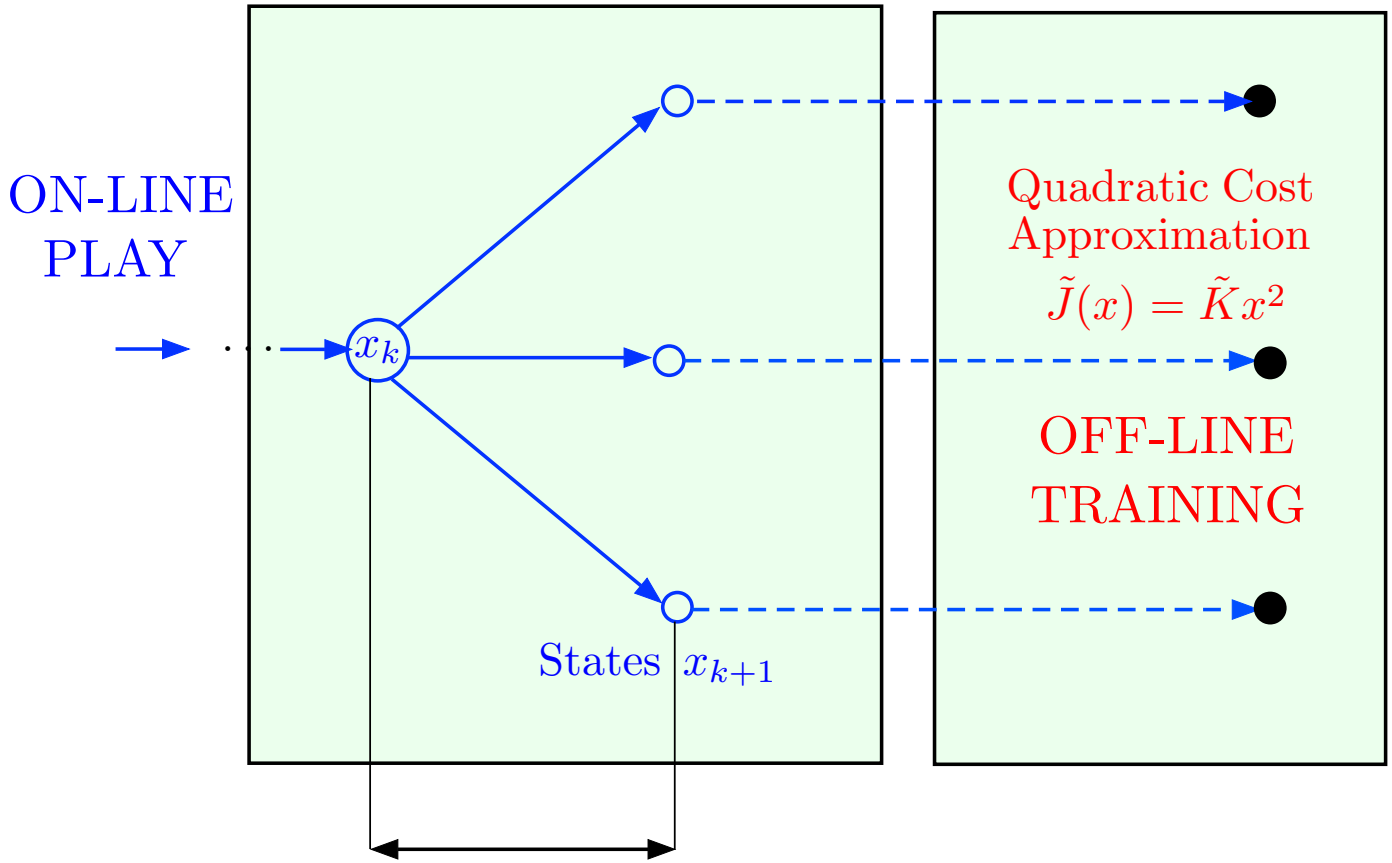
$$= \min_{L\in\Re}\min_{u=Lx}\left\{qx^2 + ru^2 + K(ax+bu)^2\right\}$$

$$= \min_{L\in\Re}\left\{q + rL^2 + K(a+bL)^2\right\}x^2$$

or

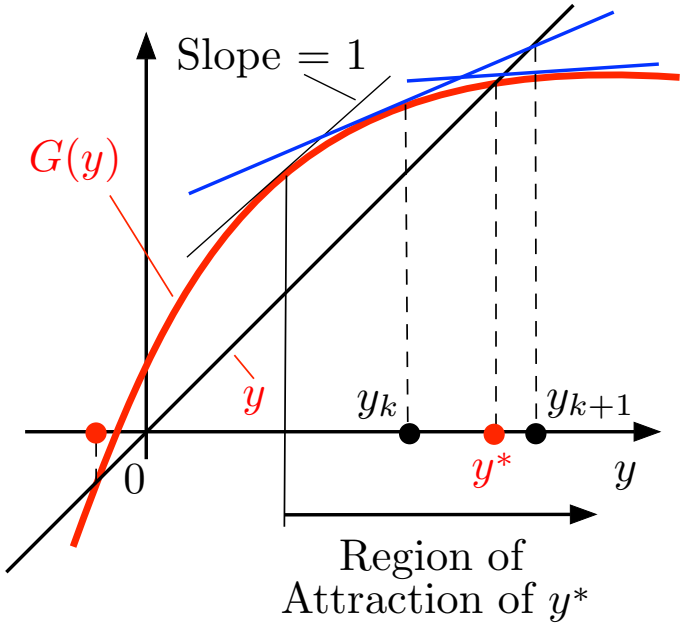
$$F(K) = \min_{L\in\Re} F_L(K), \quad \text{with} \quad F_L(K) = (a+bL)^2K + q + rL^2$$

At current state x_k, apply control $\tilde{\mu}(x_k) = \arg\min_u \{qx_k^2 + ru^2 + \tilde{K}(ax_k + bu)^2\}$



for solving the Bellman Eq. $Kx^2 = F(K)x^2$ or $K = F(K)$

Newton's Method to Solve the Generic Fixed Point Problem $y = G(y)$



At the typical iteration k

- We linearize the problem at the current iterate y_k with a first order expansion of G,

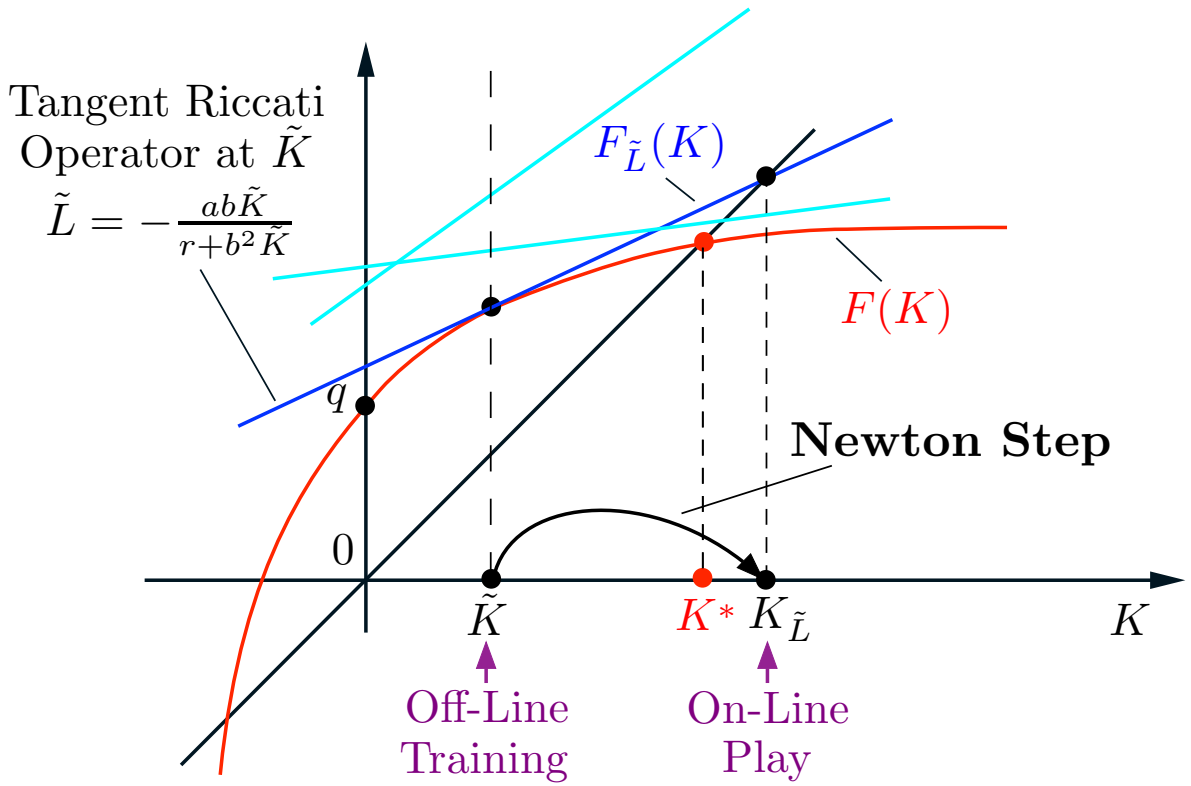
$$G(y) \approx G(y_k) + \nabla G(y_k)(y - y_k),$$

  where $\nabla G(y_k)$ is the gradient of G at y_k
- We solve the linearized problem to obtain y_{k+1}:

$$y_{k+1} = G(y_k) + \nabla G(y_k)(y_{k+1} - y_k)$$

- Extends to solution of fixed point problem $y = \min\{G_1(y), \ldots, G_m(y)\}$

Visualization of Approximation in Value Space - One-Step Lookahead - No rollout



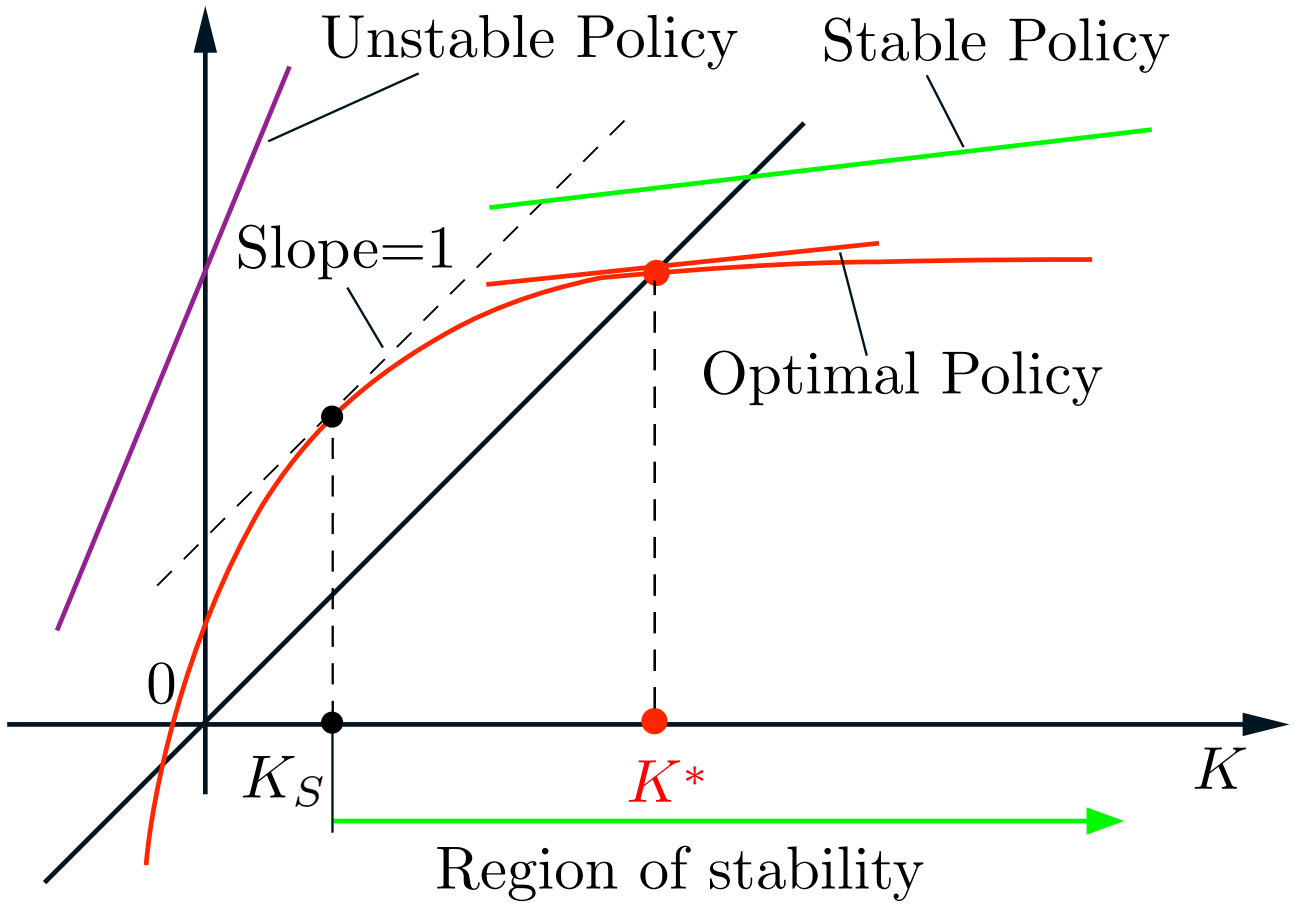
Given quadratic cost approximation $\tilde{J}(x) = \tilde{K}x^2$, we find

$$\tilde{L} = \arg\min_L F_L(\tilde{K})$$

to construct the one-step lookahead policy $\tilde{\mu}(x) = \tilde{L}x$

and its cost function $J_{\tilde{\mu}}(x) = K_{\tilde{L}}x^2$

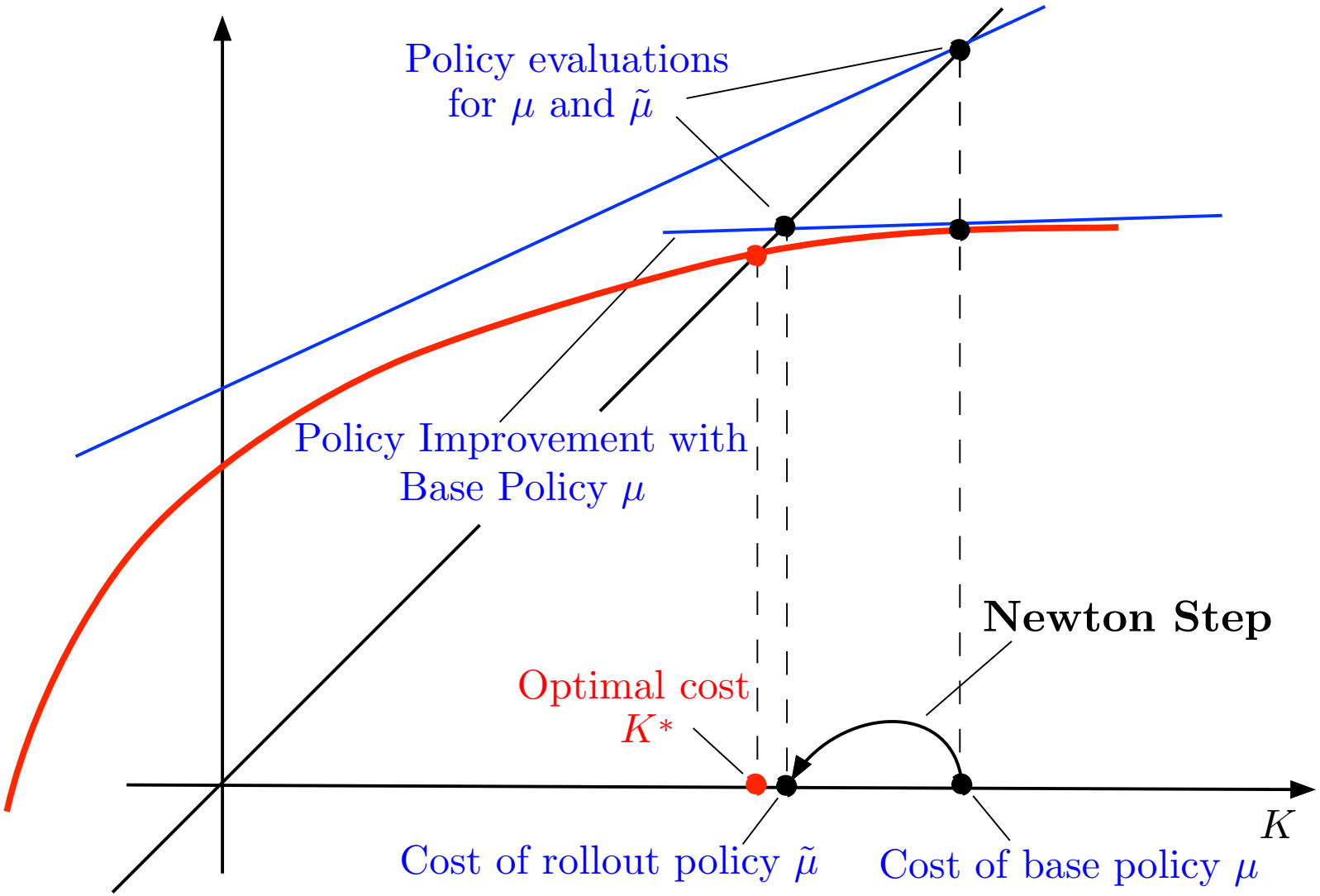
Visualization of Region of Stability of the One-Step Lookahead Policy



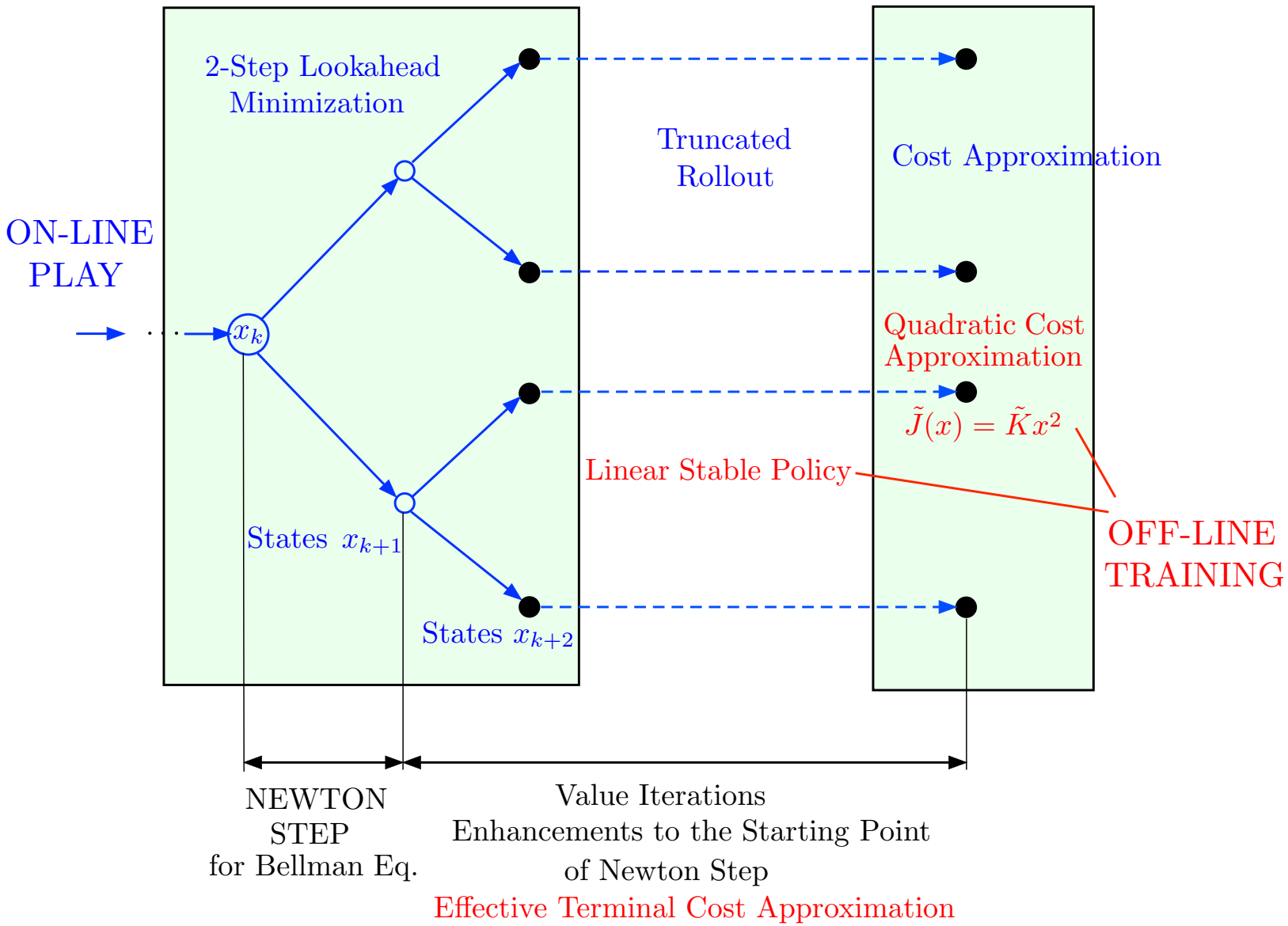
also
Region of Attraction of
Newton's Method

The start of the Newton step must be within the region of stability

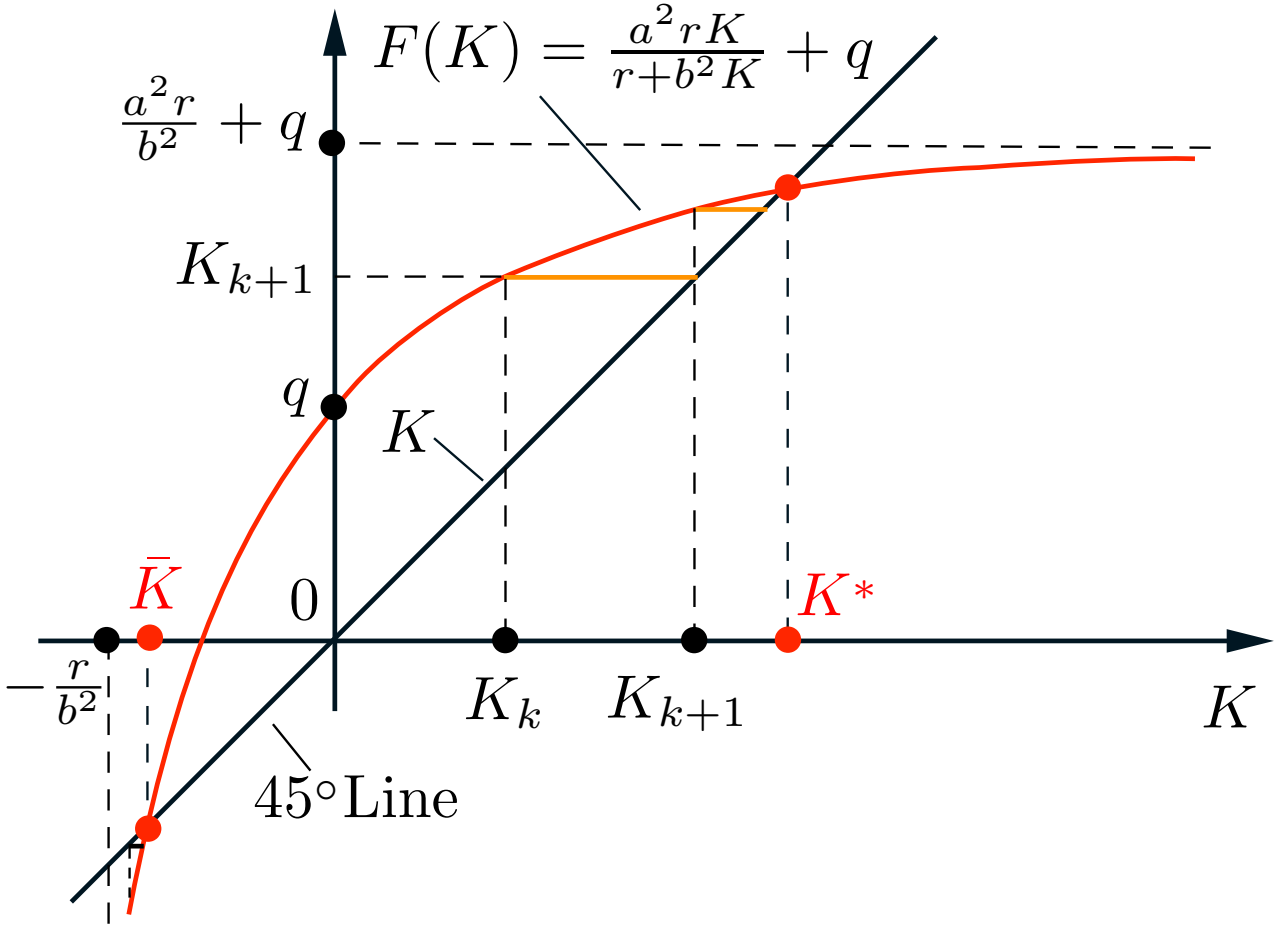
Visualization of Rollout with Stable Linear Base Policy μ: $\tilde{J} = J_\mu$



Approximation in Value Space: Multistep Lookahead



Visualization of VI

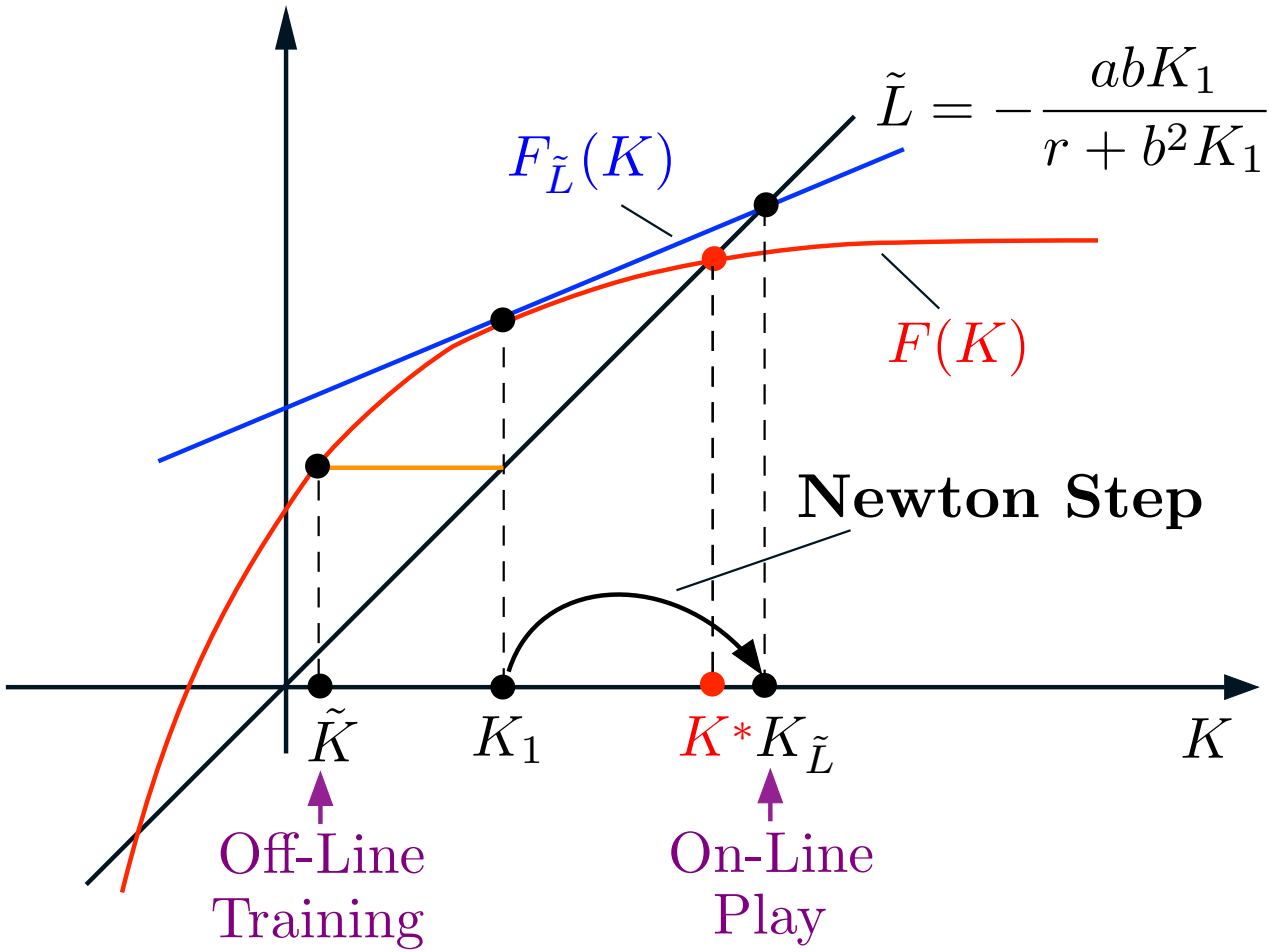


Value Iteration: $K_{k+1} = F(K_k)$

from

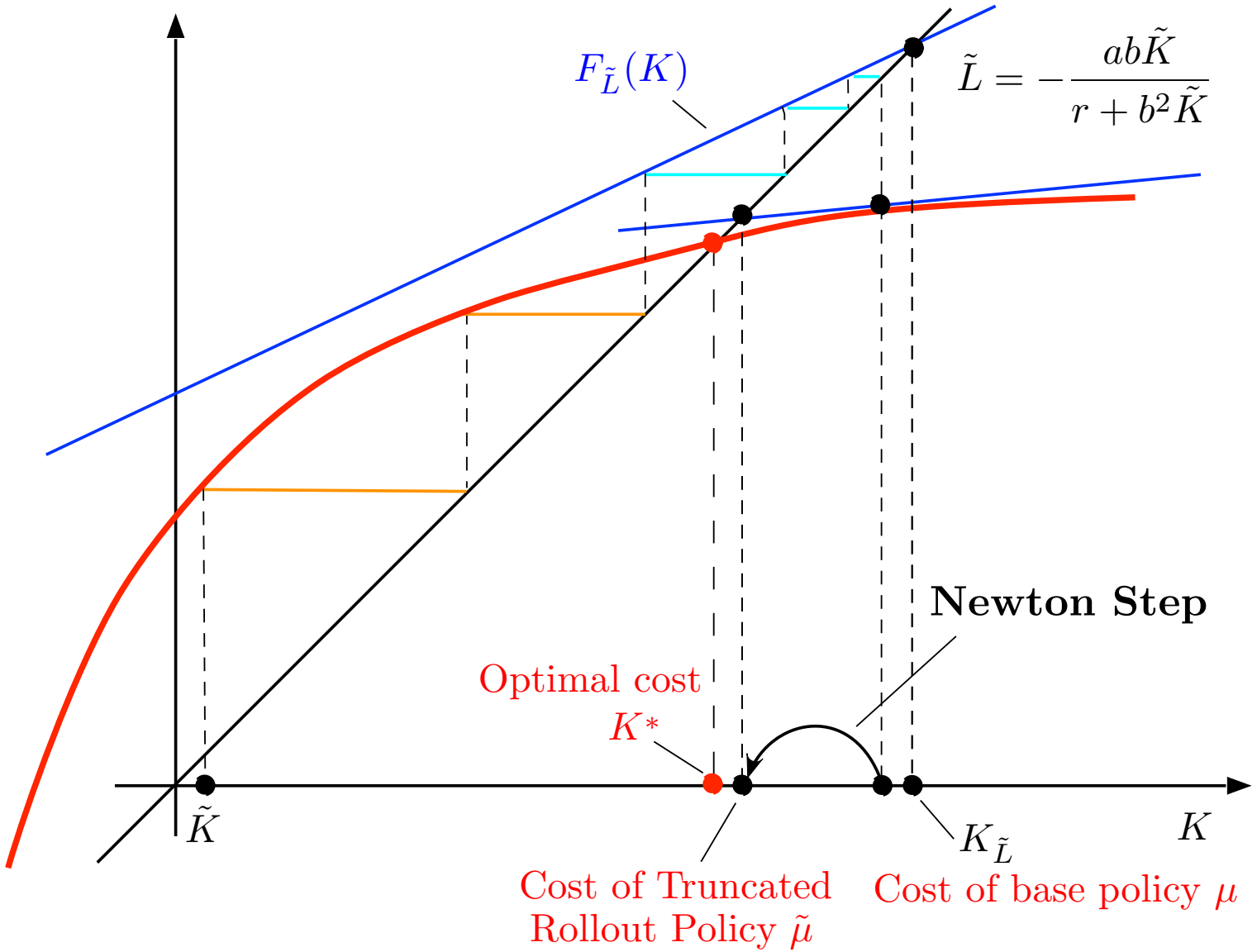
$$J_{k+1}(x) = K_{k+1}x^2 = F(K_k)x^2$$

Visualization of Approximation in Value Space - Two-Step Lookahead - No rollout



Multistep lookahead moves
the starting point of the Newton step closer to K^*
The longer the lookahead the better

Visualization of Truncated Rollout - Two-Step Lookahead - Three Truncated Rollout Steps Starting from $\tilde{K}$



Policy Iteration for the Linear Quadratic Problem (Repeated Rollout)

Starts with linear policy $\mu^0(x) = L_0 x$, generates sequence of linear policies $\mu^k(x) = L_k x$ with a two-step process

- Policy evaluation:

$$J_{\mu^k}(x) = K_k x^2$$

where

$$K_k = \frac{q + rL_k^2}{1 - (a + bL_k)^2}$$

- Policy improvement:

$$\mu^{k+1}(x) = L_{k+1} x$$

where

$$L_{k+1} = -\frac{abK_k}{r + b^2 K_k}$$

- Rollout is a single Newton iteration
- PI is a full-fledged Newton method for solving the Riccati equation $K = F(K)$
- An important variant, Optimistic PI, consists of repeated truncated rollout iterations
- Can be viewed as a Newton-SOR method (repeated application of a Newton step, preceded by first order VIs)

Generalization

The Newton step interpretation of approximation in value space generalizes very broadly

See the "Lessons from AlphaZero ..." textbook

- Riccati operators —> Bellman operators
- Newton's method for solving the min-Riccati equation —> Newton's method for solving the min-Bellman equation
- Approximation in value space is a single Newton iteration, enhanced by multistep lookahead (if any), and by truncated rollout (if any)
- Rollout is a single Newton iteration starting from the cost function of the (stable) base policy
- Exact PI is a full-fledged Newton's method
- Multistep lookahead and truncated rollout enhance the stability properties of the policy produced by approximation in value space

A Fifteen-Minute Break

Catch our breath and think about issues relating to the first half of the lecture.
Ask questions when you return.

How do we Formulate DP Problems?

An informal recipe: First define the controls, then the stages (and info available at each stage), and then the states

- Define as state x_k something that “summarizes" the past for purposes of future optimization, i.e., as long as we know x_k, all past information is irrelevant.
- Rationale: The controller applies action that depends on the state. So the state must subsume all info that is useful for decision/control.

Some examples

- In the traveling salesman problem, we need to include all the relevant info in the state (e.g., the past cities visited, and the current city). Other info, such as the costs incurred, need not be included in the state.
- In partial or imperfect information problems, we use “noisy" measurements for control of some quantity of interest y_k that evolves over time (e.g., the position/velocity vector of a moving object). It is correct to use I_k (the collection of all measurements up to time k) as state.
- It may also be correct to use alternative states; e.g., the conditional probability distribution $P_k(y_k \mid I_k)$. This is called belief state, and subsumes all the information that is useful for the purposes of control choice.

State Augmentation: Delays

$$x_{k+1} = f_k(x_k, x_{k-1}, u_k, u_{k-1}, w_k)$$

- Introduce additional state variables y_k and s_k, where $y_k = x_{k-1}$, $s_k = u_{k-1}$. Then

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \\ s_{k+1} \end{pmatrix} = \begin{pmatrix} f_k(x_k, y_k, u_k, s_k, w_k) \\ x_k \\ u_k \end{pmatrix}$$

- Define $\tilde{x}_k = (x_k, y_k, s_k)$ as the new state, we have

$$\tilde{x}_{k+1} = \tilde{f}_k(\tilde{x}_k, u_k, w_k)$$

- Reformulated DP algorithm: Start with $J_N^*(x_N) = g_N(x_N)$

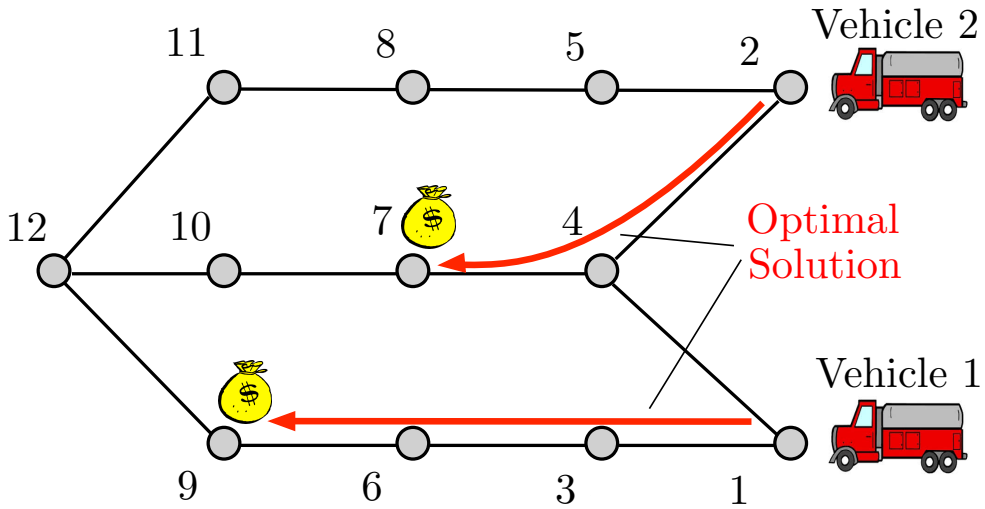
$$J_k^*(x_k, x_{k-1}, u_{k-1}) = \min_{u_k \in U_k(x_k)} E_{w_k}\left\{ g_k(x_k, u_k, w_k) + J_{k+1}^*\left(f_k(x_k, x_{k-1}, u_k, u_{k-1}, w_k), x_k, u_k\right)\right\}$$

$$J_0^*(x_0) = \min_{u_0 \in U_0(x_0)} E_{w_0}\left\{ g_0(x_0, u_0, w_0) + J_1^*\left(f_0(x_0, u_0, w_0), x_0, u_0\right)\right\}$$

See class notes for other types of state augmentation (e.g., forecasts of future uncertainty)

Problems with a Cost-Free and Absorbing Terminal State

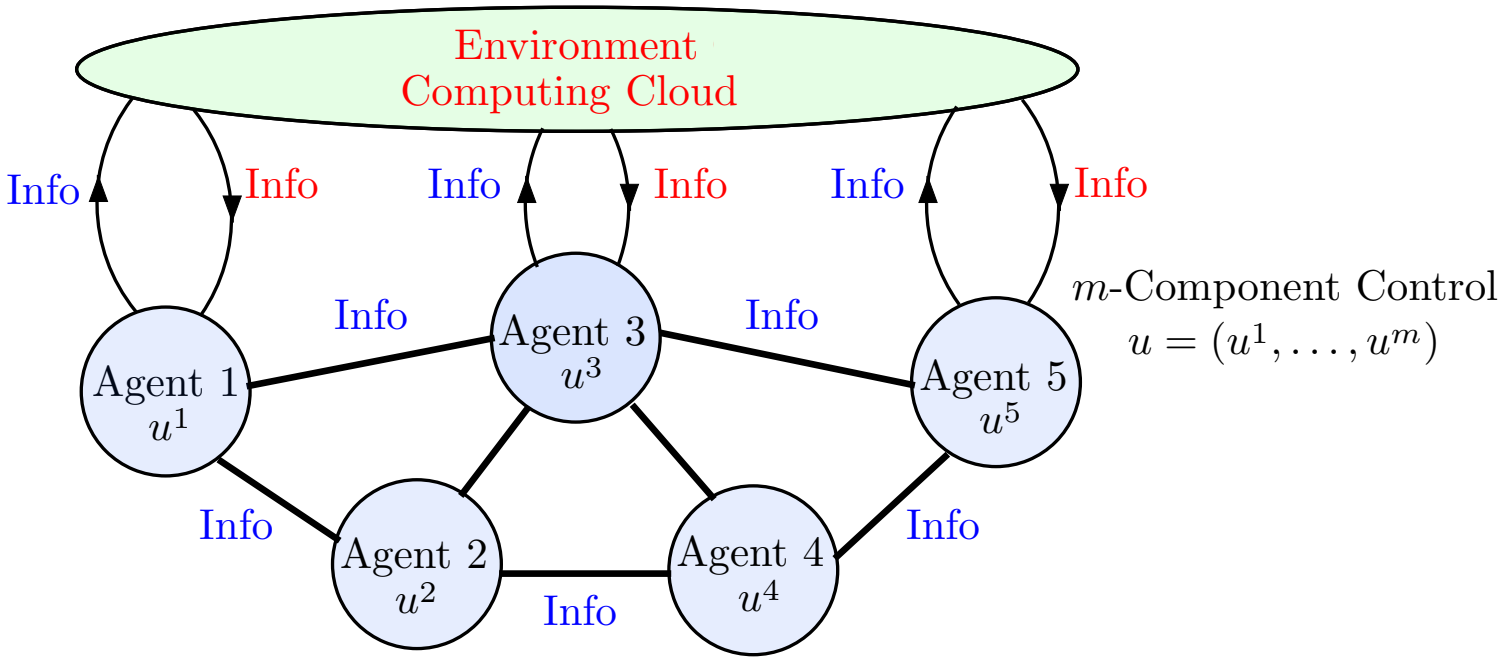
- Generally, we can view them as infinite horizon problems
- Another possibility is to convert to a finite horizon problem: Introduce as horizon an upper bound to the optimal number of stages (assuming such a bound is known)
- Add BIG penalty for not terminating before the end of the horizon



Example: Multi-vehicle routing; vehicles move one step at a time

- Minimize the number of moves to perform all tasks (i.e., reach the terminal state)
- How to formulate as DP? States? Controls? Terminal state? Horizon?
- Problem “size"? Astronomical, even for modest number of tasks and vehicles
- A good candidate for the multiagent framework to be introduced next

Multiagent Problems (1960s $\rightarrow$)

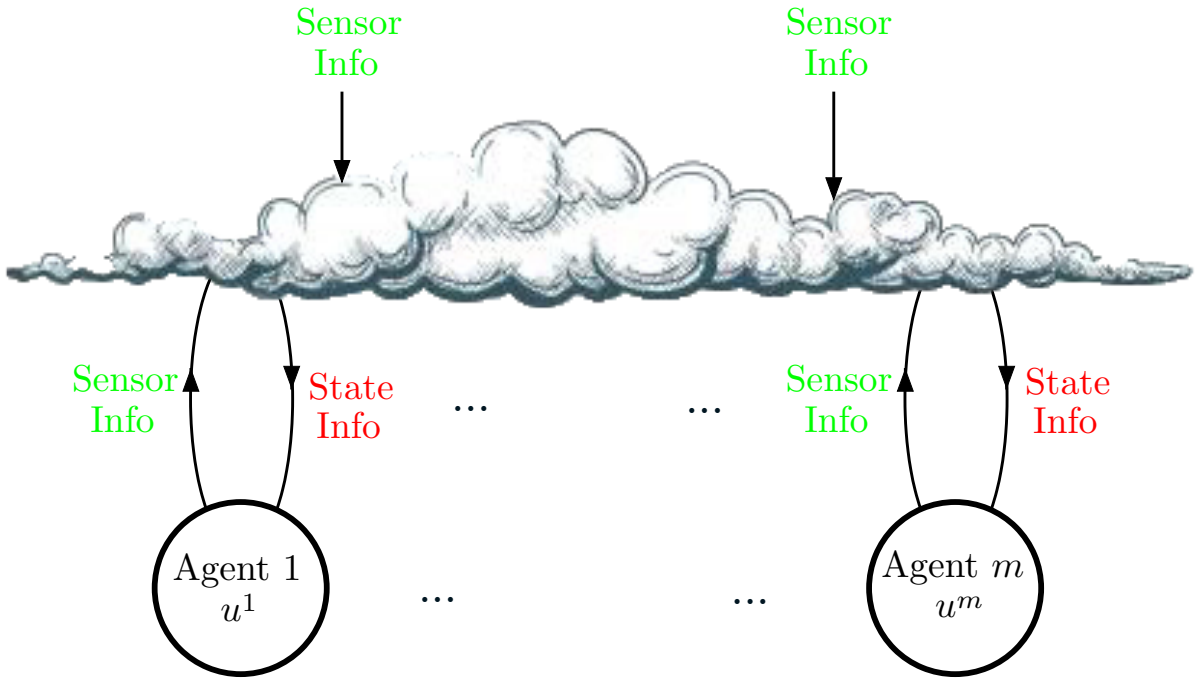


- Multiple agents collecting and sharing information selectively with each other and with an environment/computing cloud
- Agent i applies decision u^i sequentially in discrete time based on info received

The major mathematical distinction between problem structures

- The classical information pattern: Agents are fully cooperative, fully sharing and never forgetting information. Can be treated by DP
- The nonclassical information pattern: Agents are partially sharing information, and may be antagonistic. HARD because it is hard to treat by DP

Starting Point: A Classical Information Pattern (We Generalize Later)

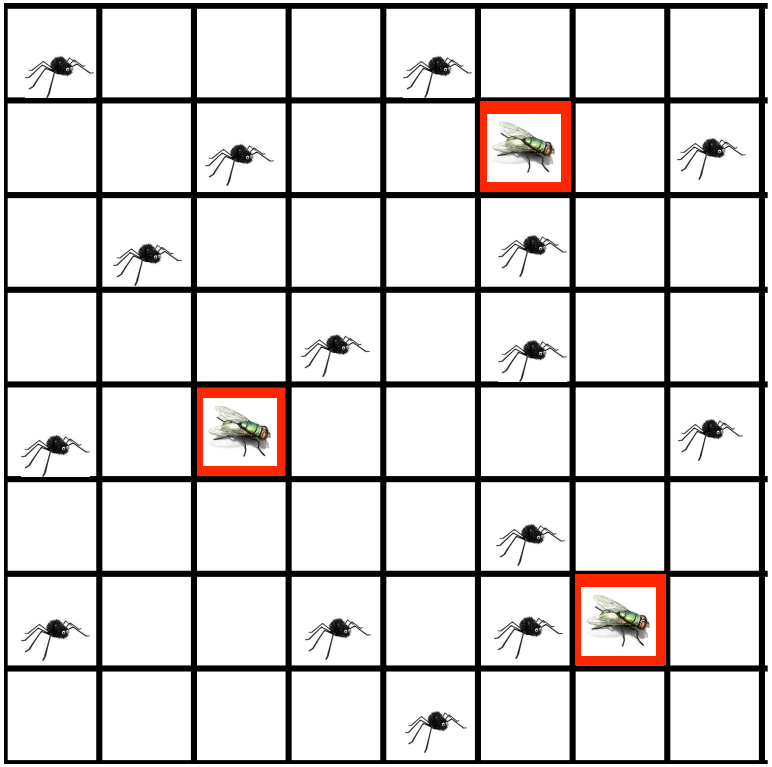


At each time: Agents have exact state info; choose their controls as function of state

Model: A discrete-time (possibly stochastic) system with state x and control u

- Decision/control has m components $u = (u^1, \ldots, u^m)$ corresponding to m "agents"
- "Agents" is just a metaphor - the important math structure is $u = (u^1, \ldots, u^m)$
- The theoretical framework is DP. We will reformulate for faster computation
  - We first aim to deal with the exponential size of the search/control space
  - Later we will discuss how to compute the agent controls in distributed fashion (in the process we will deal in part with nonclassical info pattern issues)

Spiders-and-Flies Example
(e.g., Vehicle Routing, Maintenance, Search-and-Rescue, Firefighting)



15 spiders move in 4 directions with perfect vision

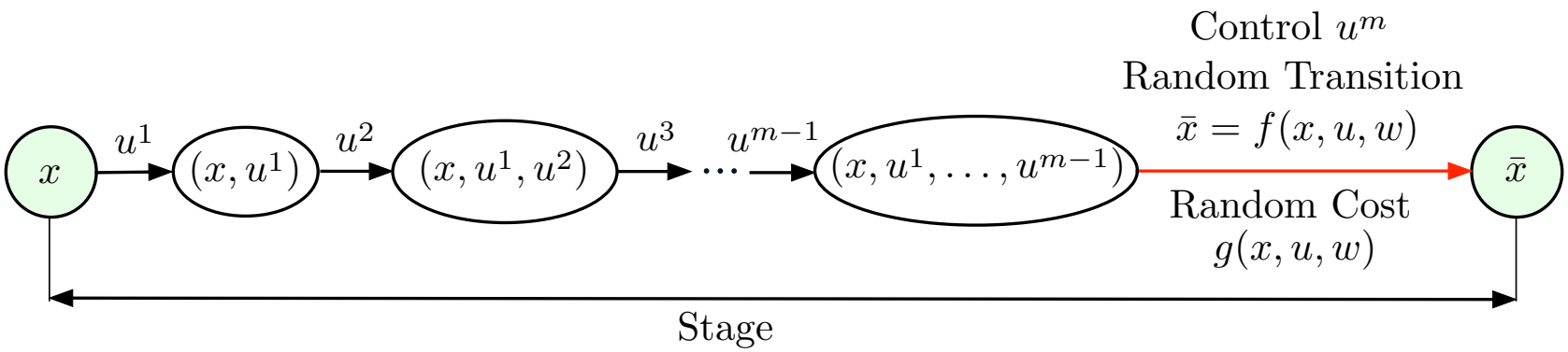
3 blind flies move randomly

Objective is to

Catch the flies in minimum time

- At each time we must select one out of $\approx 5^{15}$ joint move choices
- We will reduce to $5 \cdot 15 = 75$ (while maintaining good properties)
- Idea: Break down the control into a sequence of one-spider-at-a-time moves
- For more discussion, including illustrative videos of spiders-and-flies problems, see https://www.youtube.com/watch?v=eqbb6vVlN38&t=1654s

Reformulation Idea: Trading off Control and State Complexity



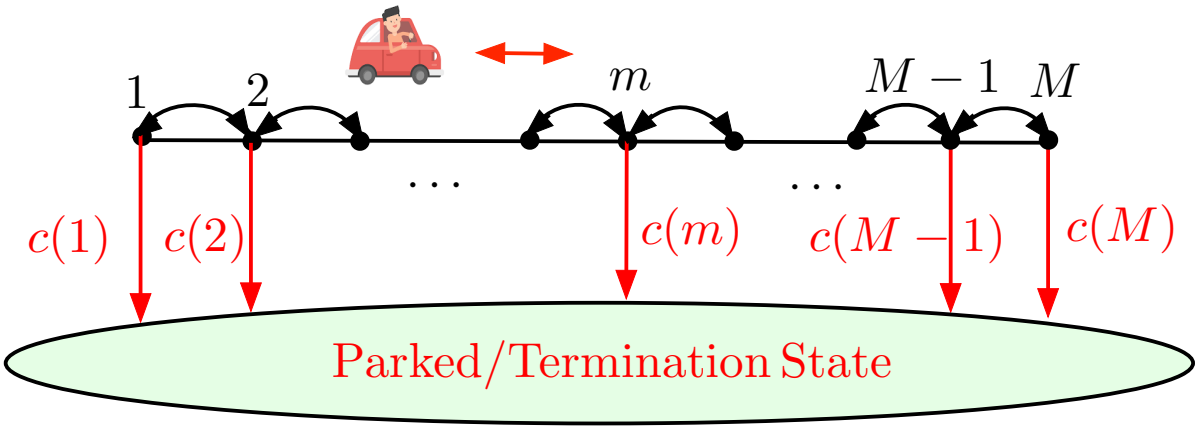
An equivalent reformulation - “Unfolding" the control action

- The control space is simplified at the expense of $m-1$ additional layers of states, and corresponding $m-1$ cost functions

$$J^1(x, u^1), J^2(x, u^1, u^2), \ldots, J^{m-1}(x, u^1, \ldots, u^{m-1})$$

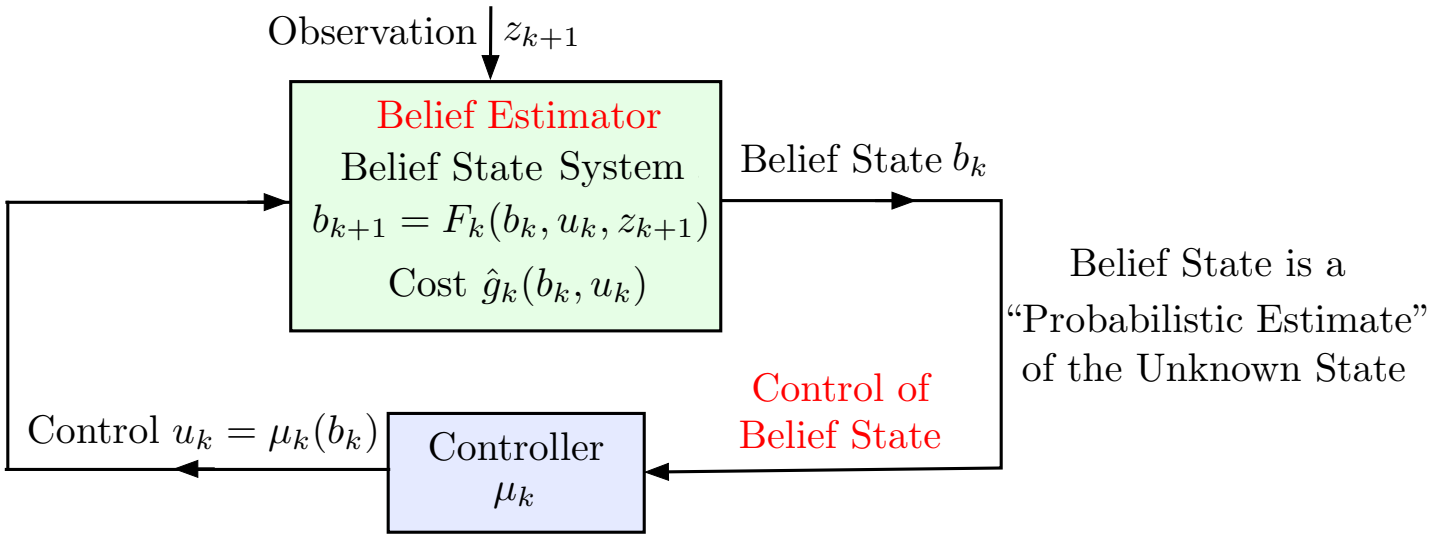
- Allows far more efficient rollout (one-agent-at-a-time). This is just standard rollout for the reformulated problem (so it involves a Newton step)
- The increase in size of the state space does not adversely affect rollout (only one state and its successors are looked at each stage during on-line play)
- Complexity reduction: The one-step lookahead branching factor is reduced from n^m to $n \cdot m$, where n is the number of possible choices for each component u^i

Parking with a Deadline: An Example of Partial State Observation



- At each time step, move one spot in either direction. Decide to park or not at spot m (if free) at cost $c(m)$. If we have not parked by time N there is a large cost C
- We observe the free/taken status of only the spot we are in. Parking spots may change status at the next time step with some probability.
- The free/taken status of the spots is “estimated" in a “probabilistic sense" based on the observations (the free/taken status of the spots visited ... when visited)
- What should the “state" be? It should summarize all the info needed for the purpose of future optimization
- First candidate for state: The set of all observations so far.
- Another candidate: The “belief state", i.e., the conditional probabilities of the free/taken status of all the spots: $p(1), p(2), \ldots, p(M)$, conditioned on all the observations so far

Partial State Observation Problems: Reformulation via Belief State



The reformulated DP algorithm has the form

$$J_k^*(b_k) = \min_{u_k \in U_k} \left[\hat{g}_k(b_k, u_k) + E_{z_{k+1}} \left\{ J_{k+1}^*\big(F_k(b_k, u_k, z_{k+1})\big) \mid b_k, u_k \right\} \right]$$

- $J_k^*(b_k)$ denotes the optimal cost-to-go starting from belief state b_k at stage k
- U_k is the control constraint set at time k
- $\hat{g}_k(b_k, u_k)$ denotes expected cost of stage k: expected stage cost $g_k(x_k, u_k, w_k)$, with distribution of (x_k, w_k) determined by b_k and the distribution of w_k
- Belief estimator: $F_k(b_k, u_k, z_{k+1})$ is the next belief state, given current belief state b_k, u_k is applied, and observation z_{k+1} is obtained

About the Next Lecture

- We will discuss adaptive and model predictive control
- We will start a more in-depth discussion of rollout

HOMEWORK 2 (DUE IN ONE WEEK): EXERCISE 1.2 OF THE CLASS NOTES

READ AHEAD CHAPTER 2 OF CLASS NOTES

This is a good time to watch the summary videolecture at
https://www.youtube.com/watch?v=A7OGgpuRnuo (1-hour version)
of the book
Lessons for AlphaZero for Optimal, Model Predictive, and Adaptive Control
Also the multiagent videolecture at
https://www.youtube.com/watch?v=eqbb6vVIN38